```
!pip install tensorflow
```

```
Requirement already satisfied: tensorflow in /usr/local/lib/python3.10/dist-packages (2.17.0)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=24.3.25 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (24.3.25)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.6
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: h5py>=3.10.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.11.0)
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (18.1.1)
Requirement already satisfied: ml-dtypes<0.5.0,>=0.3.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.4.0)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.3.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from tensorflow) (24.1)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 in /usr/local/lib/py
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.31.0)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from tensorflow) (71.0.4)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.4.0)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (4.12.2)
Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.16.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.64.1)
Requirement already satisfied: tensorboard<2.18,>=2.17 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.17.0)
Requirement already satisfied: keras>=3.2.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.4.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0
Requirement already satisfied: numpy<2.0.0,>=1.23.5 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.26.4)
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.10/dist-packages (from astunparse>=1.6.0->tensorflow) (0
Requirement already satisfied: rich in /usr/local/lib/python3.10/dist-packages (from keras>=3.2.0->tensorflow) (13.7.1)
Requirement already satisfied: namex in /usr/local/lib/python3.10/dist-packages (from keras>=3.2.0->tensorflow) (0.0.8)
Requirement already satisfied: optree in /usr/local/lib/python3.10/dist-packages (from keras>=3.2.0->tensorflow) (0.12.1)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensor
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorflow) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorflow)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorflow)
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.18,>=2.17->tensorflow
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.18,>=2.17->tensorflow
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.10/dist-packages (from werkzeug>=1.0.1->tensorboard<2.18,
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.10/dist-packages (from rich->keras>=3.2.0->tensorflow
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.10/dist-packages (from rich->keras>=3.2.0->tensorfl
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.10/dist-packages (from markdown-it-py>=2.2.0->rich->keras>=3.2.0
```

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
import tensorflow as tf
from tensorflow.keras.utils import to_categorical
import time
import seaborn as sns
import matplotlib.pyplot as plt
```

```
df = pd.read_csv('/content/Iris.csv')

df = df.drop(columns=['Id'], errors='ignore')

encoder = LabelEncoder()
df['Species'] = encoder.fit_transform(df['Species'])

X = df.drop('Species', axis=1).values
y = to_categorical(df['Species'].values)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
df_setosa = df[df['Species'] == 0].head(10)
df_versicolor = df[df['Species'] == 1].head(10)
df_virginica = df[df['Species'] == 2].head(10)
print(df_setosa)
print(df_versicolor)
print(df_virginica)
```

```
   SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm  Species
0            5.1           3.5            1.4           0.2        0
1            4.9           3.0            1.4           0.2        0
```

```
2           4.7          3.2           1.3          0.2       0
3           4.6          3.1           1.5          0.2       0
4           5.0          3.6           1.4          0.2       0
5           5.4          3.9           1.7          0.4       0
6           4.6          3.4           1.4          0.3       0
7           5.0          3.4           1.5          0.2       0
8           4.4          2.9           1.4          0.2       0
9           4.9          3.1           1.5          0.1       0
    SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm  Species
50            7.0           3.2            4.7           1.4        1
51            6.4           3.2            4.5           1.5        1
52            6.9           3.1            4.9           1.5        1
53            5.5           2.3            4.0           1.3        1
54            6.5           2.8            4.6           1.5        1
55            5.7           2.8            4.5           1.3        1
56            6.3           3.3            4.7           1.6        1
57            4.9           2.4            3.3           1.0        1
58            6.6           2.9            4.6           1.3        1
59            5.2           2.7            3.9           1.4        1
     SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm  Species
100            6.3           3.3            6.0           2.5        2
101            5.8           2.7            5.1           1.9        2
102            7.1           3.0            5.9           2.1        2
103            6.3           2.9            5.6           1.8        2
104            6.5           3.0            5.8           2.2        2
105            7.6           3.0            6.6           2.1        2
106            4.9           2.5            4.5           1.7        2
107            7.3           2.9            6.3           1.8        2
108            6.7           2.5            5.8           1.8        2
109            7.2           3.6            6.1           2.5        2
```

```python
model = Sequential([
    Dense(8, input_shape=(X_train.shape[1],), activation='relu'),
    Dense(8, activation='relu'),
    Dense(3, activation='softmax')
])

model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` arg
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```python
model_improved = Sequential([
    Dense(8, input_shape=(X_train.shape[1],), activation='relu'),
    Dense(8, activation='relu'),
    Dense(3, activation='softmax')
])

model_improved.compile(optimizer='adam',
                       loss='categorical_crossentropy',
                       metrics=['accuracy'])

history_improved = model_improved.fit(X_train, y_train, epochs=20, batch_size=10, validation_split=0.1, verbose=1)

loss_improved, accuracy_improved = model_improved.evaluate(X_test, y_test, verbose=1)
accuracy_improved *= 100
print(f'Accuracy: {accuracy_improved:.4f}')
```

```
Epoch 1/20
11/11 ━━━━━━━━━━━━━━━━━━━━ 2s 26ms/step - accuracy: 0.3661 - loss: 1.1538 - val_accuracy: 0.4167 - val_loss: 1.1244
Epoch 2/20
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.3617 - loss: 1.1097 - val_accuracy: 0.4167 - val_loss: 1.1012
Epoch 3/20
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.3461 - loss: 1.0973 - val_accuracy: 0.4167 - val_loss: 1.0767
Epoch 4/20
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.3232 - loss: 1.0720 - val_accuracy: 0.4167 - val_loss: 1.0518
Epoch 5/20
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.3064 - loss: 1.0470 - val_accuracy: 0.4167 - val_loss: 1.0236
Epoch 6/20
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.2532 - loss: 1.0188 - val_accuracy: 0.4167 - val_loss: 0.9973
Epoch 7/20
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.4052 - loss: 0.9783 - val_accuracy: 0.6667 - val_loss: 0.9693
Epoch 8/20
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.3276 - loss: 0.9764 - val_accuracy: 0.8333 - val_loss: 0.9444
Epoch 9/20
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.6146 - loss: 0.9219 - val_accuracy: 0.9167 - val_loss: 0.9185
Epoch 10/20
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7926 - loss: 0.9077 - val_accuracy: 0.9167 - val_loss: 0.8931
Epoch 11/20
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.8262 - loss: 0.8750 - val_accuracy: 0.9167 - val_loss: 0.8712
Epoch 12/20
```

```
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7989 - loss: 0.8506 - val_accuracy: 0.9167 - val_loss: 0.8480
Epoch 13/20
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.8025 - loss: 0.8322 - val_accuracy: 0.8333 - val_loss: 0.8237
Epoch 14/20
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7644 - loss: 0.7845 - val_accuracy: 0.8333 - val_loss: 0.7995
Epoch 15/20
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7639 - loss: 0.7425 - val_accuracy: 0.8333 - val_loss: 0.7754
Epoch 16/20
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7718 - loss: 0.7412 - val_accuracy: 0.8333 - val_loss: 0.7525
Epoch 17/20
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7367 - loss: 0.7144 - val_accuracy: 0.8333 - val_loss: 0.7295
Epoch 18/20
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7376 - loss: 0.7176 - val_accuracy: 0.8333 - val_loss: 0.7066
Epoch 19/20
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.7916 - loss: 0.6430 - val_accuracy: 0.8333 - val_loss: 0.6831
Epoch 20/20
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8144 - loss: 0.6013 - val_accuracy: 0.8333 - val_loss: 0.6620
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step - accuracy: 0.9000 - loss: 0.5389
Accuracy: 90.0000
```

```
loss, accuracy = model.evaluate(X_test, y_test, verbose=1)
loss = loss * 100
print(f'Test Loss: {loss:.4f}')
accuracy *= 100
print(f'Test Accuracy: {accuracy:.4f}')
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 395ms/step - accuracy: 0.3000 - loss: 1.3620
Test Loss: 136.2034
Test Accuracy: 30.0000
```

```
model_improved = Sequential([
    Dense(32, input_shape=(X_train.shape[1],), activation='relu'),
    Dense(32, activation='relu'),
    Dense(3, activation='softmax')
])

model_improved.compile(optimizer='adam',
                       loss='categorical_crossentropy',
                       metrics=['accuracy'])

history_improved = model_improved.fit(X_train, y_train, epochs=20, batch_size=10, validation_split=0.1, verbose=1)

loss_improved, accuracy_improved = model_improved.evaluate(X_test, y_test, verbose=1)
accuracy_improved *= 100
print(f'Improved Test Accuracy: {accuracy_improved:.4f}')
```

```
Epoch 1/20
11/11 ━━━━━━━━━━━━━━━━━━━━ 5s 64ms/step - accuracy: 0.7749 - loss: 0.8933 - val_accuracy: 0.7500 - val_loss: 0.9042
Epoch 2/20
11/11 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.7370 - loss: 0.7851 - val_accuracy: 0.8333 - val_loss: 0.8228
Epoch 3/20
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7516 - loss: 0.7180 - val_accuracy: 0.8333 - val_loss: 0.7557
Epoch 4/20
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7645 - loss: 0.6478 - val_accuracy: 0.8333 - val_loss: 0.6964
Epoch 5/20
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.8068 - loss: 0.5331 - val_accuracy: 0.8333 - val_loss: 0.6461
Epoch 6/20
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7836 - loss: 0.5038 - val_accuracy: 0.8333 - val_loss: 0.6067
Epoch 7/20
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.8678 - loss: 0.4026 - val_accuracy: 0.8333 - val_loss: 0.5728
Epoch 8/20
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.8777 - loss: 0.3872 - val_accuracy: 0.8333 - val_loss: 0.5425
Epoch 9/20
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.8536 - loss: 0.3873 - val_accuracy: 0.8333 - val_loss: 0.5174
Epoch 10/20
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8299 - loss: 0.3926 - val_accuracy: 0.8333 - val_loss: 0.4961
Epoch 11/20
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.8399 - loss: 0.3509 - val_accuracy: 0.9167 - val_loss: 0.4783
Epoch 12/20
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.8502 - loss: 0.3487 - val_accuracy: 0.9167 - val_loss: 0.4642
Epoch 13/20
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.8294 - loss: 0.3588 - val_accuracy: 0.9167 - val_loss: 0.4472
Epoch 14/20
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.8824 - loss: 0.2989 - val_accuracy: 0.9167 - val_loss: 0.4338
Epoch 15/20
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9007 - loss: 0.2612 - val_accuracy: 0.9167 - val_loss: 0.4216
Epoch 16/20
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.8732 - loss: 0.2838 - val_accuracy: 0.9167 - val_loss: 0.4094
Epoch 17/20
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.8861 - loss: 0.2664 - val_accuracy: 0.9167 - val_loss: 0.3957
Epoch 18/20
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.8732 - loss: 0.2764 - val_accuracy: 0.9167 - val_loss: 0.3792
Epoch 19/20
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.8864 - loss: 0.2514 - val_accuracy: 0.9167 - val_loss: 0.3688
```

```
Epoch 20/20
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9161 - loss: 0.2377 - val_accuracy: 0.9167 - val_loss: 0.3567
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step - accuracy: 0.9667 - loss: 0.1868
Improved Test Accuracy: 96.6667
```

```
import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf
from keras.datasets import mnist
```

```
from keras.datasets import mnist

(train_X, train_y), (test_X, test_y) = mnist.load_data()

print('X_train: ' + str(train_X.shape))
print('Y_train: ' + str(train_y.shape))
print('X_test:  '  + str(test_X.shape))
print('Y_test:  '  + str(test_y.shape))
```

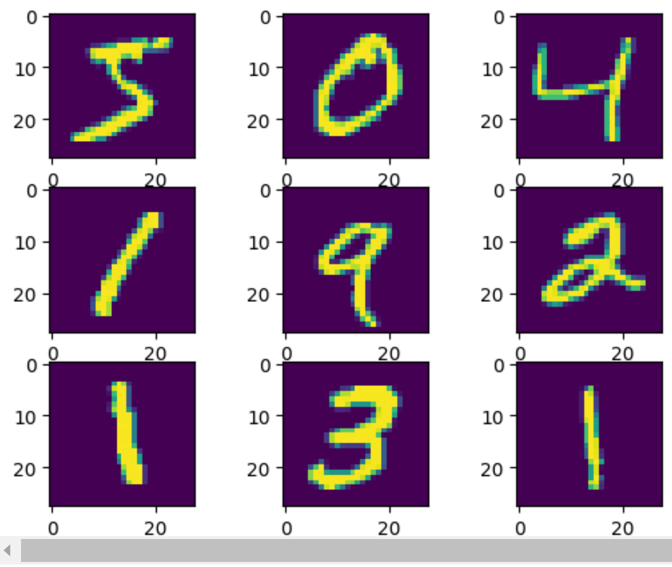
```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
X_train: (60000, 28, 28)
Y_train: (60000,)
X_test:  (10000, 28, 28)
Y_test:  (10000,)
```

```
(X_train, y_train), (X_test, y_test) = tf.keras.datasets.mnist.load_data()
```

```
print('The shape of the training inputs:', X_train.shape)
print('The shape of the training labels:',y_train.shape)
print('The shape of the testing inputs:',X_test.shape)
print('The shape of the testing labels:',y_test.shape)
```

```
The shape of the training inputs: (60000, 28, 28)
The shape of the training labels: (60000,)
The shape of the testing inputs: (10000, 28, 28)
The shape of the testing labels: (10000,)
```

```
fig, axs = plt.subplots(3, 3)
cnt = 0
for i in range(3):
    for j in range(3):
        axs[i, j].imshow(X_train[cnt])
        cnt += 1
```



```
X_train = tf.keras.utils.normalize(X_train, axis=1)
X_test = tf.keras.utils.normalize(X_test, axis=1)
```

```
model = tf.keras.models.Sequential()
```

```
model.add(tf.keras.layers.Flatten(input_shape=(28,28)))
model.add(tf.keras.layers.Dense(units=128, activation=tf.nn.relu))   # 1st hidden layer
model.add(tf.keras.layers.Dense(units=128, activation=tf.nn.relu))   # 2nd hidden layer
model.add(tf.keras.layers.Dense(units=10, activation=tf.nn.softmax))   # output layer

model.summary()
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/reshaping/flatten.py:37: UserWarning: Do not pass an `input_shape`/`input_
  super().__init__(**kwargs)
Model: "sequential_4"
```

Layer (type)	Output Shape	Param #
flatten (Flatten)	(None, 784)	0
dense_9 (Dense)	(None, 128)	100,480
dense_10 (Dense)	(None, 128)	16,512
dense_11 (Dense)	(None, 10)	1,290

```
 Total params: 118,282 (462.04 KB)
 Trainable params: 118,282 (462.04 KB)
 Non-trainable params: 0 (0.00 B)
```

```
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

```
model.fit(X_train, y_train, epochs=20, batch_size=100)
```

```
Epoch 1/20
600/600 ━━━━━━━━━━━━━━━━━━━━ 5s 5ms/step - accuracy: 0.8302 - loss: 0.6404
Epoch 2/20
600/600 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - accuracy: 0.9541 - loss: 0.1530
Epoch 3/20
600/600 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - accuracy: 0.9692 - loss: 0.1023
Epoch 4/20
600/600 ━━━━━━━━━━━━━━━━━━━━ 7s 7ms/step - accuracy: 0.9786 - loss: 0.0729
Epoch 5/20
600/600 ━━━━━━━━━━━━━━━━━━━━ 3s 5ms/step - accuracy: 0.9819 - loss: 0.0574
Epoch 6/20
600/600 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - accuracy: 0.9872 - loss: 0.0438
Epoch 7/20
600/600 ━━━━━━━━━━━━━━━━━━━━ 4s 6ms/step - accuracy: 0.9902 - loss: 0.0336
Epoch 8/20
600/600 ━━━━━━━━━━━━━━━━━━━━ 3s 6ms/step - accuracy: 0.9922 - loss: 0.0269
Epoch 9/20
600/600 ━━━━━━━━━━━━━━━━━━━━ 4s 4ms/step - accuracy: 0.9937 - loss: 0.0214
Epoch 10/20
600/600 ━━━━━━━━━━━━━━━━━━━━ 6s 5ms/step - accuracy: 0.9946 - loss: 0.0177
Epoch 11/20
600/600 ━━━━━━━━━━━━━━━━━━━━ 4s 4ms/step - accuracy: 0.9959 - loss: 0.0146
Epoch 12/20
600/600 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - accuracy: 0.9958 - loss: 0.0130
Epoch 13/20
600/600 ━━━━━━━━━━━━━━━━━━━━ 4s 6ms/step - accuracy: 0.9973 - loss: 0.0089
Epoch 14/20
600/600 ━━━━━━━━━━━━━━━━━━━━ 4s 4ms/step - accuracy: 0.9966 - loss: 0.0099
Epoch 15/20
600/600 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - accuracy: 0.9972 - loss: 0.0082
Epoch 16/20
600/600 ━━━━━━━━━━━━━━━━━━━━ 6s 6ms/step - accuracy: 0.9983 - loss: 0.0062
Epoch 17/20
600/600 ━━━━━━━━━━━━━━━━━━━━ 3s 5ms/step - accuracy: 0.9981 - loss: 0.0066
Epoch 18/20
600/600 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step - accuracy: 0.9977 - loss: 0.0063
Epoch 19/20
600/600 ━━━━━━━━━━━━━━━━━━━━ 7s 8ms/step - accuracy: 0.9988 - loss: 0.0043
Epoch 20/20
600/600 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step - accuracy: 0.9995 - loss: 0.0026
<keras.src.callbacks.history.History at 0x7aa6b9e2d0c0>
```

```
loss, accuracy = model.evaluate(X_test, y_test)
print(loss)
accuracy *= 100
print(accuracy)
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9707 - loss: 0.1477
0.1345250904560089
97.33999967575073
```

```
import tensorflow as tf
from tensorflow.keras.datasets import imdb
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense, Dropout
from tensorflow.keras.callbacks import EarlyStopping
import numpy as np

# Parameters
max_features = 10000  # Top most frequent words to consider
maxlen = 200  # Max length of review (in words)
embedding_dims = 100  # Dimension of word embedding

print("Loading data...")
(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=max_features)

print("Padding sequences...")
x_train = pad_sequences(x_train, maxlen=maxlen)
x_test = pad_sequences(x_test, maxlen=maxlen)

print("Building model...")
model = Sequential()
model.add(Embedding(max_features, embedding_dims, input_length=maxlen))
model.add(LSTM(64, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(1, activation='sigmoid'))

model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

print("Training model...")
early_stopping = EarlyStopping(monitor='val_loss', patience=3, restore_best_weights=True)

history = model.fit(x_train, y_train,
                    batch_size=32,
                    epochs=15,
                    validation_split=0.2,
                    callbacks=[early_stopping])

print("Evaluating model...")
score, acc = model.evaluate(x_test, y_test, batch_size=32)
print('Test score:', score)
print('Test accuracy:', acc)

# Make predictions on new data
def predict_sentiment(text):
    # Get the word index
    word_index = imdb.get_word_index()

    # Tokenize the new text
    words = text.lower().split()
    new_text = [word_index.get(word, 0) for word in words]

    # Pad the sequence
    new_text = pad_sequences([new_text], maxlen=maxlen)

    # Make prediction
    prediction = model.predict(new_text)
    return "Positive" if prediction[0][0] > 0.5 else "Negative"

# Example usage
sample_review = "This movie was fantastic! I really enjoyed it."
print(f"Sentiment: {predict_sentiment(sample_review)}")
```

```
Loading data...
Padding sequences...
Building model...
Training model...
Epoch 1/15
625/625 ━━━━━━━━━━━━━━━━━━━━ 93s 145ms/step - accuracy: 0.6981 - loss: 0.5607 - val_accuracy: 0.8214 - val_loss: 0.3959
Epoch 2/15
625/625 ━━━━━━━━━━━━━━━━━━━━ 91s 145ms/step - accuracy: 0.8587 - loss: 0.3430 - val_accuracy: 0.7454 - val_loss: 0.5021
Epoch 3/15
625/625 ━━━━━━━━━━━━━━━━━━━━ 91s 145ms/step - accuracy: 0.8499 - loss: 0.3539 - val_accuracy: 0.8256 - val_loss: 0.4170
Epoch 4/15
625/625 ━━━━━━━━━━━━━━━━━━━━ 143s 146ms/step - accuracy: 0.8922 - loss: 0.2720 - val_accuracy: 0.8264 - val_loss: 0.4130
Evaluating model...
782/782 ━━━━━━━━━━━━━━━━━━━━ 30s 39ms/step - accuracy: 0.8218 - loss: 0.3954
Test score: 0.39333289861679077
Test accuracy: 0.8229600191116333
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 445ms/step
Sentiment: Positive
```